Leading a Visitor Intelligently

Luke Wright, Katherine Hackworth, Stone Tejeda

Abstract

Currently, UT Austin's BWIBots are capable of navigating to 2D goals, rooms, doors, and other locales within the building. We present functionality to lead a visitor to locations throughout the GDC through an RQT-based GUI, which would allow both visitors and students the ability to navigate properly throughout the building while also getting to have simple interactions with the AI robotics research that is happening here on campus.

Introduction

In our initial proposal, we outlined five subgroups of goals for the project: the core functionality, and four additional ideas that could potentially be implemented. The first subgroup contained two key functionalities, a GUI for selecting navigation goals, and a text confirmation system that would ask the user if they were still there. Both of these we achieved, as we will explore further. The other subgroups contain goals such as announcing turns, voice control, and using a rear-facing camera or LIDAR to detect if the user is still there without manually prompting them to confirm so.

While navigation is already handled by other parts of the BWI system, several challenges must be addressed in developing a solution for leading a visitor. Most of these challenges exist largely as human-robot-interaction (HRI) issues that are more based in user experience than technical challenges. First, leading a person requires some type of confirmation that the user is still engaged and following as the robot leads them through the building. Further, that confirmation should be as unobtrusive as possible so as to minimize the annoyance to the user and help them arrive at the location as quickly as possible. Additionally, the robot should provide some method of communication to the user that the process is working as intended and they are being led successfully. If a user is to become lost, annoyed with the robot, or the task somehow otherwise fails, the robot must consider what an appropriate fallback is following failure. Though we did not address this challenge in our approach, an important consideration is in establishing a comfortable distance to lead the user from and maintaining a constant and predictable velocity and route. Finally, the safety of the user being led and all other humans must be constantly prioritized by any robot responsible for navigating and leading humans through an environment.

Our solution to address the goals and challenges of leading a user throughout the GDC was to build a graphical user interface (GUI) on top of existing navigational functionality that allows the user to select a destination in an intuitive manner, communicates the state of the navigation to the user, and confirms the user is following the robot. Should the user become separated or otherwise cancel the navigation, the robot returns to the lab. The lab was selected as a logical “home base” due to it being the location the robots are stored, charged, and tested at.

Background/Related Work

Currently, the BWIBots are fully capable of navigating around the building through an interface known as RViz, which creates a visual representation of the various floors of the GDC and allows users to send the robot to any valid xy-position on the map. Additionally, the bwi_kr_execution package contains a domain of various location labels which correspond to specific xy-coordinates on these maps, which allow further code to send the robot manually to locations without having to know the exact position of those locations through an implementation of the actionlib SimpleActionServer and SimpleActionClient. For our work, we utilized this SimpleActionClient and the existing room domain in order to allow users to specify their desired location through room names instead of actual coordinates.

Our project is not the first instance of a robot leading persons around a building. Perhaps the earliest and most prominent case of this task comes from the robot Minerva [2], which was a tour-guide robot designed by researchers at Carnegie-Mellon. Designed for use in the National Museum of American History, Minerva gave guided tours of exhibits, using speech synthesis software to describe them to visitors. Like the BWIBots, Minerva had an in-built map that was double-checked against sensory data for accuracy of movement (though in this case Minerva actually tracked the ceiling of the museum with a webcam in addition to using LIDAR). Additionally, the software controlling Minerva was comprised of many asynchronous modules that handled individual segments of Minerva’s control logic, which is very similar to how ROS nodes communicate with each other.

As Minerva was specifically designed for short-term HRI, the project was very relevant to what we implemented with the BWIBots. Something that the BWIBots sorely lack that Minerva didn’t (even in 1998), was the ability to tell people to get out of the way. If one of the BWIBots is surrounded by people, it will either try to find a path around the people (which can sometimes result in the robot taking an unrealistically long path), or simply delocalize entirely.

As far as how a robot should lead a person, we again looked to research done at Carnegie Mellon - this time for more detailed inspiration pertaining to HRI [1]. This particular robot, named Grace, used a digital face to interact with users whilst guiding them around the building, and utilized visual sensory data to maintain a specified distance away from the users so as to keep the experience as comfortable as possible. Due to time constraints and the fact that only one of our robots would be capable of using visual sensors to check for persons consistently, we opted to instead simply have the robot lead the person, and periodically check to see if the user is still following the robot to the designated goal. Additionally, since there is not an in-place humanoid face designed for the BWIBots, we opted to use a simple RQT-based GUI to interact with our users in order to maintain a simple interaction between user and robot.

Technical Approach

Core Functionality

Navigation to a navigation goal is already handled by the BWI SimpleActionServer implementation, and doors or rooms are converted from a string representation to a usable navigation goal within the domain of bwi_kr_execution. However, current implementations require setting a goal via the command line or using RViz. We implemented a basic GUI in QT that could be used by a larger range of users, whether or not they have any bash experience or are familiar enough with the BWIBots to use RViz. In the GUI we first added two hard-coded choices of rooms for goals in bwi_kr_execution for testing purposes.

The GUI Plugin

After achieving our initial selection with hard-coded buttons for various rooms we were faced with a challenge of allowing the user to select from a large number of rooms within the GDC. We made the decision that a dropdown menu would be an efficient and intuitive way for a user to select from many different room numbers.

Our initial GUI was built atop an existing RQT plugin called QuestionDialogPlugin, located in bwi_common/bwi_rqt_plugins. This plugin was copied and modified to allow the subscribing code to create a dropdown menu and listen for changes to that menu's value. The plugin is sent a list of room numbers in the format seen around the GDC (ex. 3.500) and the plugin listens for a submitted value, parses that into the format used for the navgoals in existing BWI code (ex. d3_500). The plugin then publishes the internal representation of the door, and the main lead program updates the goal to be that door. The use of python for this plugin allows this to string manipulation to be done in a relatively simple manner.

The plugin exists as a separate ROS package within our repository, but was initially developed and is currently up to date in a separate repository (https://github.com/simplyluke/lead_rqt_plugins). Our hope is that this will allow the plugin to be used for other projects in the future that need to develop a GUI with a dropdown menu, perhaps it could be merged with the existing codebase to update QuestionDialogPlugin and simply extend its functionality. Our motivation for not updating the existing source code of the BWI codebase was to prevent any disruption to other groups or existing functionality.

The final step in the guidance process is having the BWI Bot confirm whether or not the user is still following it to the goal, so as to avoid having the robot travel aimlessly across the building, and to ensure that the robot never ends up alone within the GDC unsupervised. This step involves two main tasks: stopping the current goal, and sending a new goal based upon whether or not the user is still present.

Stopping the Robot

Our first attempt to stop the robot involved the SimpleActionClient, which we used to send the navigation goals. This class has a function, cancelGoal(), which is supposed to cancel the current goal and allow the SimpleActionServer to take new goals; however, the current implementation of this function causes new goals to occasionally be preempted by the server, which can potentially result in the robot no longer being able to move. Further research into the functionality of the SimpleActionServer led to our current methodology for stopping the robot, which involves sending an empty ROS message of type actionlib::GoalID through a /move_base/cancel message to the robot. This forces the robot to stop, and after cancelling the queue of goals on the SimpleActionServer through the cancelAllGoals() method, allows for further goal sending to the robot with less fear of goals becoming preempted.

Reacting to Whether the User is Present

Once the robot has been properly stopped, we then had to determine whether or not the user is still following the robot. We decided to continue using the GUI for this task, and created a yes or no prompt which times out after waiting 10 seconds. Upon selecting “yes”, the original goal is simply re-sent through the SimpleActionClient. If “no” is selected, or if the question times out, a new goal is sent through the SimpleActionClient, which is a hard-coded goal containing the door label for the lab door closest to where the robots are housed. This would be the safest place for the robots to return to - although since this goal is hard-coded, it could be changed to any arbitrary room should future implementers of this program decide that some other place would be a better “home base” for the robots.

Example Demonstration

The general functionality of our project can be seen in the following video (https://youtu.be/lz58q7yM2ks). As can be seen, a user initially selects the room they’d like to navigate to from the dropdown menu (Figure 1), and the robot begins leading them to that destination. Throughout the navigation, the user is prompted whether or not they’re still there (Figure 2), and when they select that they are the motion towards the goal resumes. Should the user select “no” for whatever reason or the prompt times out after approximately 10 seconds, the robot will return to the lab (Figure 3) as seen in the following video (https://youtu.be/1ZoBTSW9AUw).

Conclusions

Looking back at the original project proposal, we completed the first two goals we laid out for ourselves. We already anticipated that the last few wouldn’t be realistically achievable over the course of only a few months, so for the sake of simplification, we consider the project to be a success. In the future, we foresee being able to add voice control and maintaining a ‘proper’ distance from the user, as stated before, but also focus on multi-floor navigation. As the system exists currently, passing a navigation goal to another floor simply results in no plan being generated at all. Even if a user is lost and can’t find an event or room, if the robot directs them to an elevator and asked them to press a button, it would allow leading people across the entire building from beginning to destination seamlessly. In the same vein as keeping HRI as seamless as possible, voice control is also a top priority, so that determining if the user is still there is much less interruptive.

In addition to looking forward through our previously stated future objectives, we came across a couple issues within the current BWI workspace that have also become objectives for the future. These objectives lie within the current implementation of the SimpleActionClient and SimpleActionServer. Even with our current solution, goals sending the robot back to the ‘home base’ are occasionally still preempted. Therefore, our first objective would be to better implement the client’s cancelGoal() function so that there are no issues with further goals being preempted.

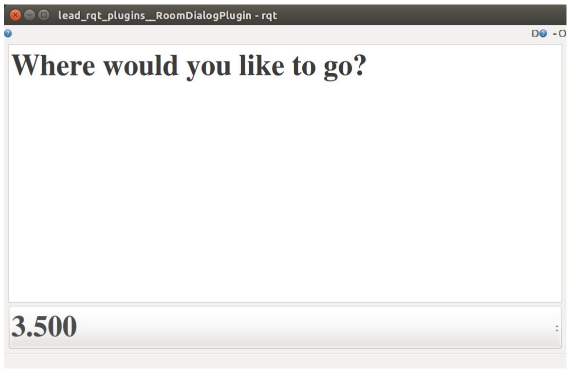
The second, and more pressing objective we have added to our list is to investigate how the current implementation of the SimpleActionServer determines whether or not a goal is successful. While our package allows the BWIBot to return to the lab upon determining that no one is following it anymore, successive returns to the lab fail due to the SimpleActionServer determining that the robot has already returned to the lab, no matter where it may actually be. We have determined that this issue occurs anytime a SimpleActionClient sends a goal which the server notices is also the most recently completed goal, and that this issue could result in problems for future projects that may utilize the SimpleActionServer. Since this server is used frequently for navigation, we have prioritized addressing this issue over the furthering of our extra proposed objectives.

Works Cited

[1] Gockley, Rachel, Jodi Forlizzi, and Reid Simmons. "Natural person-following behavior for social robots." Proceedings of the ACM/IEEE international conference on Human-robot interaction. ACM, 2007.

[2] Thrun, Sebastian, et al. "MINERVA: A second-generation museum tour-guide robot." Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on. Vol. 3. IEEE, 1999.

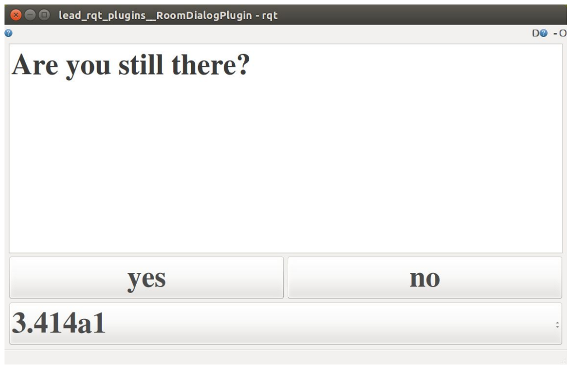
Appendix



Where would you like to go?

3.500

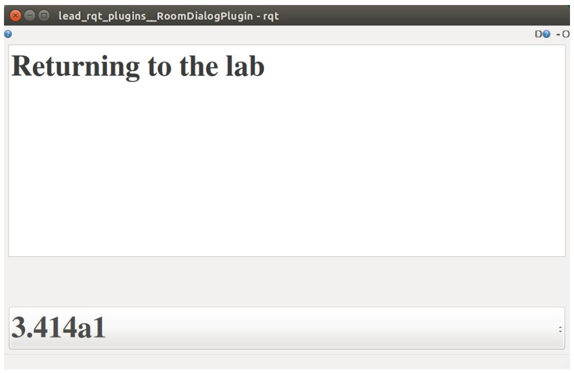
Figure 1:Initial screen, waiting for input



Are you still there?

3.414a1

Figure 2: User confirmation prompt



Returning to the lab

3.414a1

Figure 3: Fallback return screen